Package 'Fragman'

July 21, 2025

Type Package

Title Fragment Analysis in R

Version 1.0.9

Date 2018-02-01

Author Giovanny Covarrubias-Pazaran, Luis Diaz-Garcia, Brandon Schlautman, Walter Salazar, Juan Zalapa.

Maintainer Giovanny Covarrubias-Pazaran <covarrubiasp@wisc.edu>

Description Performs fragment analysis using genetic data coming from capillary electrophoresis machines. These are files with FSA extension which stands for FASTA-type file, and .txt files from Beckman CEQ 8000 system, both contain DNA fragment intensities read by machinery. In addition to visualization, it performs automatic scoring of SSRs (Sample Sequence Repeats; a type of genetic marker very common across the genome) and other type of PCR markers (standing for Polymerase Chain Reaction) in biparental populations such as F1, F2, BC (backcross), and diversity panels (collection of genetic diversity).

License GPL-3

URL http://www.wisc.edu

NeedsCompilation no

Repository CRAN

Date/Publication 2018-01-14 13:00:52 UTC

Depends R (>= 2.10)

Contents

`Fragman-package` *Fragment analysis and automatic scoring*

Description

Fragman is a package designed for Fragment analysis and automatic scoring of biparental populations (such as F1, F2, BC types) and populations for diversity studies. The program is designed to read files with FSA extension (which stands for FASTA-type file and contains lectures for DNA fragments), and .txt files from Beckman CEQ 8000 system, and extract the DNA intensities from the channels/colors where they are located, based on ABi machine plattforms to perform sizing and allele scoring.

The core of the package and the workflow of the fragment analysis rely in the following 4 functions;

1) `storing.inds`(function in charge of reading the FSA or txt(CQS) files and storing them with a list structure)

2) `ladder.info.attach` (uses the information read from the FSA files and a vector containing the ladder information (DNA size of the fragments) and matches the peaks from the channel where the ladder was run with the DNA sizes for all samples. Then loads such information in the R environment for the use of posterior functions)

3) `overview2` (create friendly plots for any number of individuals specified and can be used to design panels (`overview2`) for posterior automatic scoring (like licensed software does), or make

manual scoring (overview) of individuals such as parents of biparental populations or diversity populations)

4) The score.markers (function score the alleles by finding the peaks provided in the panel (if provided), otherwise returns all peaks present in the channel). Thisfinal function can be automatized if several markers are located in the same channel by creating lists of panels taking advantage of R capabilities and data structures.

** Sometimes during the ladder sizing process some samples can go wrong for several reasons related to the sample quality (low intensity in ladder channel, extreme number of noisy peaks, etc.), because of that we have introduced ladder.corrector function which allows the user to correct the bad samples by clicking over the real peaks, by default the ladder.info.attach function returns the names of the samples that had a low correlation with the expected peaks.

When automatic scoring is not desired the function overview can be used for getting an interactive session and click over the peaks (using the locator function) in order to get the allele sizes.

Contact

Feel free to contact us with questions and improvement suggestions at:

covarrubiasp@wis.edu

Just send a sample file with your question to recreate the issue or bug reported along with vector for your ladder.

Citation

We have spent valuable time developing this package, please cite it in your publication:

Covarrubias-Pazaran G, Diaz-Garcia L, Schlautman B, Salazar W, Zalapa J. Fragman: An R package for fragment analysis. 2016. BMC Genetics 17(62):1-8.

Author(s)

Giovanny Covarrubias-Pazaran, Luis Diaz-Garcia, Brandon Schlautman, Walter Salazar, Juan Zalapa.

References

Covarrubias-Pazaran G, Diaz-Garcia L, Schlautman B, Salazar W, Zalapa J. Fragman: An R package for fragment analysis. 2016. BMC Genetics 17(62):1-8.

Robert J. Henry. 2013. Molecular Markers in Plants. Wiley-Blackwell. ISBN 978-0-470-95951-0.

Ben Hui Liu. 1998. Statistical Genomics. CRC Press LLC. ISBN 0-8493-3166-8.

See Also

http://cggl.horticulture.wisc.edu/home-page/

Examples

```
## ================================= ##
## ================================= ##
## Fragment analysis requires
## 1) loading your data
## 2) matching your ladder
## 3) define a panel for scoring
## 4) score the samples
## ================================= ##
## ================================= ##

#####################
## 1) Load your data
#####################

### you would use something like:
# folder <- "~/myfolder"
# my.plants <- storing.inds(folder)
### here we just load our sample data and use the first 2 plants

?my.plants
data(my.plants)
my.plants <- my.plants[1:2]
class(my.plants) <- "fsa_stored"
# plot(my.plants) # to visualize the raw data

#######################
## 2) Match your ladder
#######################

### create a vector indicating the sizes of your ladder and do the match

my.ladder <- c(50, 75, 100, 125, 129, 150, 175, 200, 225, 250, 275, 300, 325, 350, 375)
ladder.info.attach(stored=my.plants, ladder=my.ladder)

### matching your ladder is a critical step and should only happen once per batch of
### samples read

#############################################################################################
### OPTIONAL:
### If the ladder.info attach function detects some bad samples
### that you can correct them manually using
### the ladder.corrector() function
### For example to correct one sample in the previous data
### ladder.corrector(stored=my.plants,
#to.correct="FHN152-CPN01_01A_GH1x35_152-148-209_717-704-793_367-382-381.fsa",
#ladder=my.ladder)
#############################################################################################

#######################
## 3) Define a panel
#######################
```

```
### In fragment analysis you usually design a panel where you indicate
### which peaks are real. You may use the overview2 function which plots all the
### plants in the channel you want in the base pair range you want

overview2(my.inds=my.plants, channel = 2:3, ladder=my.ladder, init.thresh=5000)

### You can click on the peaks you think are real, given that the ones
### suggested by the program may not be correct. This can be done by using the
### 'locator' function and press 'Esc' when you're done, i.e.:
# my.panel <- locator(type="p", pch=20, col="red")$x
### That way you can click over the peaks and get the sizes
### in base pairs stored in a vector named my.panel

### Just for demonstration purposes I will use the suggested peaks by
### the program using overview2, which will return a vector with
### expected DNA sizes to be used in the next step for scoring
### we'll do it in the 160-190 bp region

my.panel <- overview2(my.inds=my.plants, channel = 3,
                    ladder=my.ladder, init.thresh=7000,
                    xlim=c(160,190)); my.panel

##########################
## 4) Score the samples
##########################

### When a panel is created is time to score the samples by providing the initial
### data we read, the ladder vector, the panel vector, and our specifications
### of channel to score (other arguments are available)

### Here we will score our samples for channel 3 with our panel created previously

res <- score.markers(my.inds=my.plants, channel = 3, panel=my.panel$channel_3,
                ladder=my.ladder, electro=FALSE)

### Check the plots and make sure they were scored correctly. In case some samples
### are wrong you might want to use the locator function again and figure out
### the size of your peaks. To extract your peaks in a data.frame do the following:

final.results <- get.scores(res)
final.results
```

`arrange.jm` *Arrange data converted to joinmap code into a joinmap readable file*

Description

This function converts a data frame containing the joinmap code into the readable file for joinmap. This format still needs some extra information, specifically the header indicating the population type, no.loci and no. of individuals, please check file examples included in JoinMap software.

Usage

```
arrange.jm(x, par=FALSE)
```

Arguments

x	A data frame containing the scores in jm coding.
par	A TRUE/FALSE value indicating if the data returned should include the parents or not, the dafault value is FALSE, indicating that the first 2 individuals will not be included.

Details

No major details.

Value

If arguments are correct the function returns a new data frame

joinmap A new data frame with markers in joinmap readable format

References

Covarrubias-Pazaran G, Diaz-Garcia L, Schlautman B, Salazar W, Zalapa J. Fragman: An R package for fragment analysis. 2016. BMC Genetics 17(62):1-8.

Robert J. Henry. 2013. Molecular Markers in Plants. Wiley-Blackwell. ISBN 978-0-470-95951-0.

Ben Hui Liu. 1998. Statistical Genomics. CRC Press LLC. ISBN 0-8493-3166-8.

Examples

```
xx <- data.frame(cbind(a=rep(150, 96), b=c(rep(100,48), rep(150,48)))); xx[1,] <- c(150,150)
xx2 <- cbind(jm.conv(xx), jm.conv(xx), jm.conv(xx))
xx3 <- arrange.jm(xx2, par=FALSE)
xx3[,1:10]
```

---

best.layout *complementary tools for layout*

Description

This function just find the best layout fit for a number of plots desired.

Usage

```
best.layout(x)
```

Arguments

x	A scalar value indicating the number of plots desired

Details

No major details

Value

Returns the best layout

res the number of rows and columns giving the best fit

References

Covarrubias-Pazaran G, Diaz-Garcia L, Schlautman B, Salazar W, Zalapa J. Fragman: An R package for fragment analysis. 2016. BMC Genetics 17(62):1-8.

Robert J. Henry. 2013. Molecular Markers in Plants. Wiley-Blackwell. ISBN 978-0-470-95951-0.

Ben Hui Liu. 1998. Statistical Genomics. CRC Press LLC. ISBN 0-8493-3166-8.

Examples

```
best.layout(9)
```

---

big.peaks.col *Peak search by first derivatives*

Description

This function find all peaks by taking the first derivative based on 'rle' function. Is used in different Fragma functions.

Usage

```
big.peaks.col(x,tre)
```

Arguments

x	A vector of heights or intensities
tre	A scalar value deciding when peaks will be ignored

Details

No major details.

Value

Retuns the biggest peaks for a vector of intensities.

out a vector of positions where the derivative is zero and therefore a peak was found

References

Covarrubias-Pazaran G, Diaz-Garcia L, Schlautman B, Salazar W, Zalapa J. Fragman: An R package for fragment analysis. 2016. BMC Genetics 17(62):1-8.

Robert J. Henry. 2013. Molecular Markers in Plants. Wiley-Blackwell. ISBN 978-0-470-95951-0.

Ben Hui Liu. 1998. Statistical Genomics. CRC Press LLC. ISBN 0-8493-3166-8.

Examples

```
data(my.plants)
big.peaks.col(my.plants[[1]][,1],100)#for any color
```

`detect.ladder` *Ladder detection by correlation or confidence intervals*

Description

This function takes a vector of color heights/intensities from the fragment analysis containing the ladder/standard channel, and detects the biggest peaks where the derivative is equal zero and uses the information from the expected weights for the ladder to construct confidence intervals in order to detect the ladder peaks.

Please! if using the confidence interval method ("ci"), which is NOT the default, once you have found the best parameters for the arguments to match your ladder using this function, please pass those values to all the posterior functions, making sure the 'dev' argument is passed to the new functions. If using the correlation method ("cor"), don't worry about it.

Usage

```
detect.ladder(stored, ind=1, ladder, channel.ladder=dim(stored[[1]])[2],
              ci.upp=1.96, ci.low=1.96, draw=TRUE, dev=50, warn=TRUE,
            init.thresh=250,sep.index=8, method="cor", avoid=1500, who="sample")
```

Arguments

`stored`	Lis of dataframes obtained by using the `storing.inds` function.
`ind`	The individual that you wish to analyze for assessing that the ladder was correctly detected.
`ladder`	Vector containing the expected weights of the dna fragments of the ladder in use
`channel.ladder`	A scalar value indicating in which channel or color the ladder was read
`ci.upp`	A scalar value indicating how many standar errors will be used to detect peaks when checking the height of the ladder peaks(upper bound). To be used in the `find.ladder` function
`ci.low`	A scalar value indicating how many standar errors will be used to detect peaks when checking the height of the ladder peaks(lower bound). To be used in the 'find.ladder' function

draw	A TRUE/FALSE value indicating if the plot for the ladder found should be printed or not
dev	A scalar value indicating the number of indexes to be used as peak separation when deciding the ladder peaks. Some ladders contain dna fragments of very closed weights and modifying this parameter helps to detect them correctly
warn	A TRUE/FALSE value indicating if warnings should be provided when detecting the ladder
init.thresh	An initial value of color intensity to be used when detecting the ladder, could be really important for the correlation method
sep.index	A scalar value indicating how many indexes should be allowed to considered a true peak from noisy peaks
method	An argument indicating one of the 2 methods available; "cor" makes all possible combination of peaks and searches exhaustive correlations to find the right peaks corresponsding to the expected DNA weights, or "ci" constructing confidence intervals to look for peaks meeting the conditions specified in the previous arguments
who	A name to indicate which sample is being analyzed
avoid	A scalar value indicating how many indexes should be avoided when the method of correlation fails to find peaks and a random sample will be drawn from the existing peaks. The default is 1500 indexes which will samples peaks avoiding the first 1500 indexes which is usually related to noisy area in some ladders.

Details

The peaks are detected by default using a correlation method bu the user can use confidence intervals if desired.

Value

If parameters are indicated correctly the function returns:

$pos the index positions for the intensities

$hei the intensities for the fragments found

$wei the putative weights in base pairs based on the ladder provided

References

Covarrubias-Pazaran G, Diaz-Garcia L, Schlautman B, Salazar W, Zalapa J. Fragman: An R package for fragment analysis. 2016. BMC Genetics 17(62):1-8.

Robert J. Henry. 2013. Molecular Markers in Plants. Wiley-Blackwell. ISBN 978-0-470-95951-0.

Ben Hui Liu. 1998. Statistical Genomics. CRC Press LLC. ISBN 0-8493-3166-8.

Examples

```
data(my.plants)
my.ladder <- c(120, 125, 129, 150, 175, 200, 225, 250, 275, 300, 325, 350, 375)
# looking at the first individual
detect.ladder(stored=my.plants, ind=1, ladder=my.ladder)
```

`find.ladder` *Ladder detection by correlation or confidence intervals*

Description

This function takes a vector of color heights/intensities from the fragment analysis containing the ladder/standard channel, and detects the biggest peaks where the derivative is equal zero and uses the information from the expected weights for the ladder to construct confidence intervals in order to detect the ladder peaks.

Please! if using the confidence interval method ("ci"), which is not the default, once you have found the best parameters for the arguments to match your ladder using this function, please pass those values to all the posterior functions, please make sure the 'dev' argument is passed to the new functions.

Usage

```
find.ladder(x, ladder, draw=TRUE, dev=50, warn=TRUE, init.thresh=NULL,
            sep.index=8, method=NULL, reducing=NULL, who="sample",
            attempt=10, cex.title=0.8)
```

Arguments

`x`	Vector of heights from the ladder channel. See example to see how to access to it.
`ladder`	Vector containing the expected weights of the dna fragments of the ladder in use
`draw`	A TRUE/FALSE value indicating if the plot for the ladder found should be printed or not
`dev`	A scalar value indicating the number of indexes to be used as peak separation when deciding the ladder peaks. Some ladders contain dna fragments of very closed weights and modifying this parameter helps to detect them correctly
`warn`	A TRUE/FALSE value indicating if warnings should be provided when detecting the ladder
`init.thresh`	An initial value of color intensity to be used when detecting the ladder
`sep.index`	A scalar value indicating how many indexes should be allowed to considered a true peak from noisy peaks
`method`	An argument indicating one of the 2 methods available; "cor" makes all possible combination of peaks and searches exhaustive correlations to find the right peaks corresponsding to the expected DNA weights, or "ci" constructing confidence intervals to look for peaks meeting the conditions specified in the previous arguments
`who`	A name to indicate which sample is being analyzed
`attempt`	A scalar value indicating how many attempts should be made to find the real ladder peaks. By default is 7 attempts, which means that will try to build the model assuming that the first peak found in the ladder is the corresponding first

peak of the expected ladder, then moves to the 2nd peak until the 7th and the seven models are compared picking the most likely model based on the R2 value for each of the models.

`reducing` A vector of values to reduce the search of peaks to certain indexes in the x axis. Default is NULL so it looks for all peaks for matching the ladder.

`cex.title` A scalar value indicating how big the title (name of the sample) in the plot should be.

Details

We have implemented 3 methods for sizing the ladder, each with their advantages and disadvantages. The default method named "red" which stands for "reduction" detect the region where peaks exist (in indexes) in the ladder channel and assumes that your ladder should have some equivalence in indexes and creates an 'expected ladder', then the putative ladder moves along the peak region and correlations and squared distances to the closest peaks are calculated. We have define the coefficient of similarity (CS) as cor(x,y)/var(z), where:

cor(x,y) are the correlations between expected and observed peaks, and var(z) is the sum of squares between the differences of expected and observed peaks.

This value usually let us identify the most likely peaks and then all possible combinations for those peaks are computed followed by exhaustive correlations of those combinations with the actual ladder. The highest correlation usually points to the right peaks, which is selected.

In addition the method "cor" is the previous version to "red" which doesn't reduce the search of peaks and computes all possible combinations of peaks from the beggining, with the drawback that slows down the detection process especially when the ladder intensities are low and noisy peaks exist in abundance.

The last method that has been superseded by the previous 2 is the "ci" method based on confidence intervals, which assumes that real ladder peaks have more or less the same intensity and a they can be found by finding the median intensity and computing a 90 percent confidence interval to find the rest of the peaks. This method has been proved to fail when the first condition is broken and ladder have real peaks with intensities greater than the expected.

Value

If parameters are indicated correctly the function returns:

$pos the index positions for the intensities

$hei the intensities for the fragments found

$wei the putative weights in base pairs based on the ladder provided

References

Covarrubias-Pazaran G, Diaz-Garcia L, Schlautman B, Salazar W, Zalapa J. Fragman: An R package for fragment analysis. 2016. BMC Genetics 17(62):1-8.

Robert J. Henry. 2013. Molecular Markers in Plants. Wiley-Blackwell. ISBN 978-0-470-95951-0.

Ben Hui Liu. 1998. Statistical Genomics. CRC Press LLC. ISBN 0-8493-3166-8.

Examples

```
data(my.plants)
my.ladder <- c(50, 75, 100, 125, 129, 150, 175, 200, 225, 250, 275, 300, 325, 350, 375)
find.ladder(my.plants[[1]][,4], ladder=my.ladder)
```

`get.scores` *complementary tools*

Description

This function extracts the information from auto.score and score.easy functions and fits everything in a dataframe.

Usage

```
get.scores(my.scores, mark = "mark")
```

Arguments

`my.scores`	List of individuals which contains at the same time a list of positions, heights and weights for the fragment analysis
`mark`	A vector of names of the markers scored in the panel or the parents

Details

Nomajor details.

Value

If arguments are correct the function returns a data frame containing

$da A dataframe containing the ssr calls

References

Covarrubias-Pazaran G, Diaz-Garcia L, Schlautman B, Salazar W, Zalapa J. Fragman: An R package for fragment analysis. 2016. BMC Genetics 17(62):1-8.

Robert J. Henry. 2013. Molecular Markers in Plants. Wiley-Blackwell. ISBN 978-0-470-95951-0.

Ben Hui Liu. 1998. Statistical Genomics. CRC Press LLC. ISBN 0-8493-3166-8.

Examples

```
## here "a" is a similar ouptput to the `score.easy` function
par1 <- list(pos=c(3100, 3240), hei=c(22917,20563), wei=c(202,212))
par2 <- list(pos=c(3100, 3240), hei=c(22917,20563), wei=c(202,214))
a <- list(i1=par1, i2=par2)
get.scores(a)
```

homo.panel *complementary tools*

Description

This functions takes a list of positions, heights and weights for ssr calls of a certain plant and uses panel information and a window to homogenize the weights of base pairs to the panel calls provided.

Usage

```
homo.panel(x, panel, window)
```

Arguments

x	list of positions, heights and weights
panel	different dna sizes usually obtained by using overview and locator functions
window	window in base pairs indicating when should be considered the same panel allele

Details

No major details.

Value

If arguments are correct the function returns a list containing

$pos the index positions for the intensities

$hei the intensities for the fragments found

$wei the putative weights in base pairs based on the panel provided

References

Robert J. Henry. 2013. Molecular Markers in Plants. Wiley-Blackwell. ISBN 978-0-470-95951-0.

Ben Hui Liu. 1998. Statistical Genomics. CRC Press LLC. ISBN 0-8493-3166-8.

Examples

```
#No example provided, internally working to round to the closest parent ssr call.
x <- 1:10
```

`homogenize.to.parentals` *complementary tools*

Description

This functions takes a list of positions, heights and weights for ssr calls of a certain plant and uses parental information and a window to homogenize to the parent calls.

Usage

```
homogenize.to.parentals(x, parents, window)
```

Arguments

`x`	list of positions, heights and weights
`parents`	different parental calls
`window`	window in base pairs indicating when should be considered the same allele

Details

No major details.

Value

If arguments are correct the function returns a list containing

$pos the index positions for the intensities

$hei the intensities for the fragments found

$wei the putative weights in base pairs based on the ladder provided

References

Robert J. Henry. 2013. Molecular Markers in Plants. Wiley-Blackwell. ISBN 978-0-470-95951-0.

Ben Hui Liu. 1998. Statistical Genomics. CRC Press LLC. ISBN 0-8493-3166-8.

Examples

```
#No example provided, internally working to round to the closest parent ssr call.
x <- 1:10
```

jm.conv *Scores to JoinMap converter*

Description

This function converts a data frame containing the scores from score.easy to a new data frame with joinmap calls. The parents need to be provided in the first and second row. Currently only works for CP type of crosses. From CP type to F2 and BC2 is straight forward.

Usage

```
jm.conv(a)
```

Arguments

a	A data frame containing the scores from score.easy function extracted from get.scores function containing the parental calls in the first 2 rows.

Details

No major details.

Value

If arguments are correct the function returns a new data frame

res A new data frame with markers in joinmap format

References

Covarrubias-Pazaran G, Diaz-Garcia L, Schlautman B, Salazar W, Zalapa J. Fragman: An R package for fragment analysis. 2016. BMC Genetics 17(62):1-8.

Robert J. Henry. 2013. Molecular Markers in Plants. Wiley-Blackwell. ISBN 978-0-470-95951-0.

Ben Hui Liu. 1998. Statistical Genomics. CRC Press LLC. ISBN 0-8493-3166-8.

Examples

```
xx <- data.frame(cbind(a=rep(150, 96), b=c(rep(100,48), rep(150,48))))
jm.conv(xx)
# try using apply to a dataframe
```

`ladder.corrector` *Ladder corrector attached to R environment*

Description

This function was designed to correct manually samples that could not be correctly detected by the `ladder.info.attach` function and allows the user to select manually the peaks he knows are the correct peaks. This function uses the output of the `ladder.info.attach` function which is basically a vector with the names of the samples that were too dificult for the algorithm to find. The console will draw a plot and will ask the user to click over the peaks expected for a ladder provided, once the user is done should press the 'esc' key and continue to the next sample. This process is repeated until the all samples with the names provided are adjusted.

Usage

```
ladder.corrector(stored, to.correct, ladder,
      thresh=200, env = parent.frame(),...)
```

Arguments

`stored`	List with the channels information from the individuals specified, usually coming from the `storing.inds` function output.
`to.correct`	Vector containing the names ofthe samples to be corrected and usually the output of the `ladder.info.attach` function.
`ladder`	Vector containing the expected weights of the dna fragments of the ladder in use.
`thresh`	A scalar value indicating the minimum value in RFUs to look for peaks to match the user-selected peaks with the program recognized peaks.
`env`	this is used to detect the environment of the user and load the result in the same environment.
`...`	Further arguments to be passed

Details

Once the user has selected the right peaks the function will fix the ladder and attach such information to the R environment.

Value

This function does not produce any output.

$data the program will attach the information to the R environment

References

We have spent valuable time developing this package, please cite it in your publication:

Covarrubias-Pazaran G, Diaz-Garcia L, Schlautman B, Salazar W, Zalapa J. Fragman: An R package for fragment analysis. 2016. BMC Genetics 17(62):1-8.

Robert J. Henry. 2013. Molecular Markers in Plants. Wiley-Blackwell. ISBN 978-0-470-95951-0.

Ben Hui Liu. 1998. Statistical Genomics. CRC Press LLC. ISBN 0-8493-3166-8.

Examples

```
data(my.plants)
my.plants <- my.plants[1:2]
my.ladder <- c(50, 75, 100, 125, 129, 150, 175, 200, 225, 250, 275, 300, 325, 350, 375)
ladder.info.attach(stored=my.plants, ladder=my.ladder, ladd.init.thresh=300)
## now if something goes wrong use the corrector:
#ladder.corrector(stored=my.plants,
#to.correct="FHN152-CPN01_01A_GH1x35_152-148-209_717-704-793_367-382-381.fsa",
#ladder=my.ladder)
```

---

ladder.info.attach *Ladder detection and attachment to R environment*

Description

This function uses the information stored by the `storing.inds` function and a vector specifying the ladder/standard and finds the real peaks corresponding to the expected weights. The user may use this function to be able to load the ladder information in the global environment of R, so when using the `overview` or `score.markers` functions calculations will be performed faster, if the function is not used the program will calculate the ladder information each time `overview` or `score.markers` functions are used.

NOTE: THE STEP OF MATCHING THE LADDER WITH YOUR SAMPLES USING THE ‘ladder.info.attach‘ FUNCTION IS CRITICAL. IF YOU HAVE ANY PROBLEM TRY MODIFYING THE ARGUMENT ’method’, WITH THE 2 MOST EFFECTIVE METHODS method="iter" OR method="iter2"

Usage

```
ladder.info.attach(stored, ladder, channel.ladder=NULL,
                   method="iter2", ladd.init.thresh=NULL,
                   env = parent.frame(), prog=TRUE,
                   draw=TRUE, attempt=10)
```

Arguments

`stored`	List with the channels information from the individuals specified, usually coming from the `storing.inds` function output.

`ladder` Vector containing the expected weights of the dna fragments of the ladder in use.

`channel.ladder` A scalar value indicating in which channel or color the ladder was read

`method` An argument indicating one of the methods available; "iter" makes an iterative procedure to find the ladder, "iter2" the same but backwards, "cor" makes all possible combination of peaks and searches exhaustive correlations to find the right peaks corresponsding to the expected DNA weights, or "ci" constructing confidence intervals to look for peaks meeting the conditions specified in the previous arguments. The default allows the program to pick among "iter" and "iter2". Older methods have been depreciated.

`ladd.init.thresh` A value of intensity to detect peaks in the internal use of the `find.ladder` function. We recommend not to deal to much with it unless you identified special situations with your ladder

`env` this is used to detect the environment of the user and load the result in the same environment.

`prog` A TRUE/FALSE value indicating if a progress bar should be drawn while processing the samples in order to assess the time it takes to find the ladder. The dafault value is TRUE but usually this makes the process slower. Please feel free to set it equal FALSE if the number of samples is quite large and speed is a concern.

`draw` A TRUE/FALSE value indicating if a plot showing the peaks matched with your ladder should be drawn. The dafault value is FALSE to avoid extra delay. Please feel free to set it equal TRUE if you prefer to assess the sizing process.

`attempt` A scalar value indicating how many attempts should be made to find the real ladder peaks when using the "iter" and "iter2" methods. By default is 7 attempts, which means that will try to build the model assuming that the first peak found in the ladder is the corresponding first peak of the expected ladder, then moves to the 2nd peak until the 7th and the seven models are compared picking the most likely model based on the R2 value for each of the models.

Details

Method "ci" has been depreciated, currently the method "iter2" is the default and uses the ladder provided and observed peaks to match them using an iterative procedure based on least squares.

Value

If parameters are indicated correctly the function returns:

$pos the index positions for the intensities

$hei the intensities for the fragments found

$wei the putative weights in base pairs based on the ladder provided

References

We have spent valuable time developing this package, please cite it in your publication:

Covarrubias-Pazaran G, Diaz-Garcia L, Schlautman B, Salazar W, Zalapa J. Fragman: An R package for fragment analysis. 2016. BMC Genetics 17(62):1-8.

Robert J. Henry. 2013. Molecular Markers in Plants. Wiley-Blackwell. ISBN 978-0-470-95951-0.

Ben Hui Liu. 1998. Statistical Genomics. CRC Press LLC. ISBN 0-8493-3166-8.

Examples

```
data(my.plants)
my.plants <- my.plants[1:2]
my.ladder <- c(50, 75, 100, 125, 129, 150, 175, 200, 225, 250, 275, 300, 325, 350, 375)
ladder.info.attach(stored=my.plants, ladder=my.ladder)
```

---

lapply_pb *complementary tools for Fragman*

Description

This function is a wrapper of lapply function that allows the drawing of a progress bar to assess the speed of the process.

Usage

```
lapply_pb (X, FUN, ...)
```

Arguments

X	a vector (atomic or list) or an expression object. See see lapply.
FUN	the function to be applied to each element of X: see lapply.
...	passes another arguments to the typical lapply function.

Details

No major details

Value

Performs lapply drawing a progress bar

res the same result than using lapply

References

See see lapply

Examples

```
l <- sapply(1:200, function(x) list(rnorm(1000)))
lapply_pb(l, mean)
```

`letter.to.jm` *Letter to JoinMap code converter*

Description

This function converts a vector of ssr calls in letter format to joinmap code using information from mother and father provided in first and second row respectively

Usage

```
letter.to.jm(x)
```

Arguments

`x`	A vector of ssr calls in letter format or snp types, mother and father of the population should be in 1st and 2nd position respectively

Details

If numeric data exists first needs to be converted to letter code in order to use this function.

Value

If arguments are correct the function returns a list containing

$y A vector with ssr calls in joinmap format

References

Covarrubias-Pazaran G, Diaz-Garcia L, Schlautman B, Salazar W, Zalapa J. Fragman: An R package for fragment analysis. 2016. BMC Genetics 17(62):1-8.

Robert J. Henry. 2013. Molecular Markers in Plants. Wiley-Blackwell. ISBN 978-0-470-95951-0.

Ben Hui Liu. 1998. Statistical Genomics. CRC Press LLC. ISBN 0-8493-3166-8.

Examples

```
xx <- data.frame(cbind(a=rep(150, 96), b=c(rep(100,48), rep(150,48))))
xx1 <- num.to.lett(xx)
letter.to.jm(unlist(xx1))
# try using apply to a dataframe
```

my.plants *Cranberry biparental population*

Description

This dataset are 60 individuals from a progeny coming from the cross of 2 cranberry plants. Six SSR markers were run, 2 in the first channel (blue), 2 in the second channel (green), 2 in the third channel (yellow) and the Roxtrash375 ladder was run in the fourth channel (red).

Usage

```
data("my.plants")
```

Format

The format is: chr "my.plants"

Details

The data is basically the raw FSA files coming from the ABi machine. No more details for this data.

Source

This data was generated by the Cranberry Genomics Lab.

References

Covarrubias-Pazaran G, Diaz-Garcia L, Schlautman B, Salazar W, Zalapa J. Fragma: An R package for fragment analysis. http://horticulture.wisc.edu/cggl/ZalapaLab/People.html. 2015.

Examples

```
data(my.plants)
## look at the list structure
str(my.plants)
```

num.to.lett *Number to Letter code converter*

Description

This function converts dataframes with rounded calls (numeric format in 2 cells), to letter format based on the GBS pipeline developed by Elshire et al. (2011) which can be used as intermediate step to transform to joinmap and Onemap formats, it requires mother and father in first and second row

Usage

```
num.to.lett(xx)
```

Arguments

xx	matrix with numbers, every 2 columns is a marker and each row is an individual, parents are located in the first 2 rows

Details

No major details.

Value

If arguments are correct the function returns a list containing

xx2 matrix coded in letter format where each column is a marker and each row is an individual

References

Covarrubias-Pazaran G, Diaz-Garcia L, Schlautman B, Salazar W, Zalapa J. Fragman: An R package for fragment analysis. 2016. BMC Genetics 17(62):1-8.

Robert J. Henry. 2013. Molecular Markers in Plants. Wiley-Blackwell. ISBN 978-0-470-95951-0.

Ben Hui Liu. 1998. Statistical Genomics. CRC Press LLC. ISBN 0-8493-3166-8.

Examples

```
xx <- data.frame(cbind(a=rep(150, 96), b=c(rep(100,48), rep(150,48))))
num.to.lett(xx)
```

---

overview *Assesing several plants with an overview*

Description

This function uses information from the FSA files read from storing.inds function and creates a plot to assess graphically the peaks of several plants in certain channel in order to score manually or assess the parental fragments in the case of biparentla ppulations. If you desire to create a panel you may want to take a look at overview2. The function contains several defaults in most of the arguments, please check arguments but in general.

Usage

```
overview(my.inds, channel = 1, n.inds = c(1:length(my.inds)),
     xlimi=c(min(ladder),max(ladder)), ladder, channel.ladder=dim(my.inds[[1]])[2],
        ploidy=2, dev=50, method="iter",
        init.thresh=200, ladd.init.thresh=200, warn=TRUE, my.palette=NULL,
        env = parent.frame())
```

Arguments

`my.inds`	List with the channels information from the individuals specified, usually coming from the `storing.inds` function output
`channel`	The channel you wish to analyze, usually 1 is blue, 2 is green, 3 is yellow, 4 is red and so on
`n.inds`	Vector specifying the plants to be scored
`xlimi`	A vector containing the base pair interval where the plot should be drawn
`ladder`	A vector containing the expected weights for the ladder peaks that will be found the using the `find.ladder` function
`channel.ladder`	A scalar value indicating in which channel or color the ladder was read
`ploidy`	A scalar value indicating the ploidy of the organism to be scored
`dev`	A scalar value indicating the number of indexes to be used as peak separation when deciding the ladder peaks, for more details check `find.ladder` function
`method`	An argument indicating one of the 2 methods available; "cor" makes all possible combination of peaks and searches exhaustive correlations to find the right peaks corresponsding to the expected DNA weights, or "ci" constructing confidence intervals to look for peaks meeting the conditions specified in the previous arguments
`init.thresh`	An initial value of intensity to detect peaks. We recommend not to deal to much with unless you have highly controlled dna concentrations in your experiment
`ladd.init.thresh`	A value of intensity to detect peaks in the internal use of the `find.ladder` function. We recommend not to deal to much with it unless you identified special situations with your ladder
`warn`	A TRUE/FALSE value indicating if warnings should be provided when detecting the ladder
`my.palette`	A character vector with the colors to be used when drawing the RFU plots. If NULL it will use the programmed palette.
`env`	this is used to detect the environment of the user and load the result in the same environment.

Details

No major details.

Value

If rarguments are correct the function returns a list containing

$plot Returns a plot joining the channel for the plants specified for the color desired and the peaks found by the function using the parameters specified

$nana Returns a vector with the names of the plants specified in the function

References

Covarrubias-Pazaran G, Diaz-Garcia L, Schlautman B, Salazar W, Zalapa J. Fragman: An R package for fragment analysis. 2016. BMC Genetics 17(62):1-8.

Robert J. Henry. 2013. Molecular Markers in Plants. Wiley-Blackwell. ISBN 978-0-470-95951-0.

Ben Hui Liu. 1998. Statistical Genomics. CRC Press LLC. ISBN 0-8493-3166-8.

Examples

```
data(my.plants)
my.plants <- my.plants[1:10]
my.ladder <- c(50, 75, 100, 125, 129, 150, 175, 200, 225, 250, 275, 300, 325, 350, 375)
overview(my.inds=my.plants, channel = 1, n.inds = c(1:5), ladder=my.ladder, xlim=c(200,220))
# now use:
# locator(type="p", pch=20, col="red")$x
# to click over the peaks and get the sizes in base pairs
# when you are done make sure you press the "Esc" key,
# do not push the stop button, some versions of R usually crash
# by stopping instead of pressing 'Esc'.
```

`overview2` *Assesing several plants with an overview*

Description

This function uses information from the FSA files read from `storing.inds` function and creates an overlapping plot to assess graphically the peaks of several plants in certain channel in order to create a panel for the scoring functions `score.markers`. The function contains several defaults in most of the arguments, please check arguments but in general you only need the first 4 arguments to create a panel.

Usage

```
overview2(my.inds, channel = 1, ladder, xlim = NULL, ylim = NULL,
        n.inds = NULL, channel.ladder = NULL, ploidy = 2,
        method="iter2", init.thresh=NULL, ladd.init.thresh=200,
        lwd=.25, warn=TRUE, min.panel=100, suggested=TRUE,
        env = parent.frame(), my.palette=NULL, verbose=TRUE)
```

Arguments

`my.inds`	List with the channels information from the individuals specified, usually coming from the `storing.inds` function output
`channel`	The channel/color you wish to analyze, usually 1 is blue, 2 is green, 3 is yellow, 4 is red and so on
`ladder`	A vector containing the expected weights for the ladder peaks that will be found the using the `find.ladder` function

`xlim`	A vector containing the base pair interval where the plot should be drawn
`ylim`	A vector containing the intensity interval where the plot should be drawn
`n.inds`	Vector specifying the plants to be scored
`channel.ladder`	A scalar value indicating in which channel or color the ladder was read
`ploidy`	A scalar value indicating the ploidy of the organism to be scored
`method`	An argument indicating one of the 2 methods available; "cor" makes all possible combination of peaks and searches exhaustive correlations to find the right peaks corresponsding to the expected DNA weights, or "ci" constructing confidence intervals to look for peaks meeting the conditions specified in the previous arguments
`init.thresh`	An initial value of intensity to detect peaks. We recommend not to deal to much with unless you have highly controlled dna concentrations in your experiment
`ladd.init.thresh`	A value of intensity to detect peaks in the internal use of the `find.ladder` function. We recommend not to deal to much with it unless you identified special situations with your ladder
`lwd`	The width of the line
`warn`	A TRUE/FALSE value indicating if warnings should be provided when detecting the ladder
`min.panel`	A scalar value indicating which peak values should be ignored when creating a panel. If 'xlim' values are specified the 'min.panel' value is ignored and instead the panel peaks provided by the program are based in the region where you are zooming in.
`suggested`	a TRUE/FALSE value statement declaring if you want the program to return suggested peaks for your panel. The default is TRUE but can be anoying if the program draws too many peaks.
`env`	this is used to detect the environment of the user and load the result in the same environment.
`my.palette`	A character vector with the colors to be used when drawing the RFU plots. If NULL it will use the programmed palette.
`verbose`	A TRUE/FALSE statement indicating if the function should return informative messages.

Details

No major details.

Value

If rarguments are correct the function returns a list containing

$plot Returns a plot joining the channel for the plants specified for the color desired and the peaks found by the function using the parameters specified

$nana Returns a vector with the names of the plants specified in the function

References

We have spent valuable time developing this package, please cite it in your publication:

Covarrubias-Pazaran G, Diaz-Garcia L, Schlautman B, Salazar W, Zalapa J. Fragman: An R package for fragment analysis. 2016. BMC Genetics 17(62):1-8.

Robert J. Henry. 2013. Molecular Markers in Plants. Wiley-Blackwell. ISBN 978-0-470-95951-0.

Ben Hui Liu. 1998. Statistical Genomics. CRC Press LLC. ISBN 0-8493-3166-8.

Examples

```
data(my.plants)
my.plants <- my.plants[1]
my.ladder <- c(50, 75, 100, 125, 129, 150, 175, 200, 225, 250, 275, 300, 325, 350, 375)
overview2(my.inds=my.plants, channel = 1, ladder=my.ladder, lwd=1)
# now use:
# my.panel <- locator(type="p", pch=20, col="red")$x
# to click over the peaks and get the sizes in base pairs
# when you are done make sure you press the "Esc" key, do not push the stop button

## to look at many channels at the same time you
## can use the par(new=TRUE) and a for loop

for(u in 1:4){
  overview2(my.inds=my.plants, channel = u, ladder=my.ladder, lwd=1,
            xlim=c(240,350), ylim=c(0,30000))
  par(new=TRUE)
}
```

---

plot.fsa_stored *plot form fsa files stored with storing.inds*

Description

plot method for class "fsa_stored".

Usage

```
## S3 method for class 'fsa_stored'
plot(x, lay=c(2,1), channel=NULL, cex.legend=.5, ncol.legend=4,lims=NULL, color=NULL, ...)
```

Arguments

x	an object of class "fsa_stored"
lay	layout for the numberof plots to visualize.
channel	if preferred a single channel can be specified.
cex.legend	value of the size of the legend.

ncol.legend	number of columns to divide the legend.
lims	equivalen to ylim argument in plot, xlim still can be used.
color	specific colors to use in case the user don't want the default palete.
...	Further arguments to be passed

Value

vector of plot

Author(s)

Giovanny Covarrubias <covarrubiasp@wisc.edu>

See Also

Fragman

---

pullup *Applying pullup to channels/colors*

Description

This function takes a matrix of DNA intensities and merge all the channels (columns) to identify overall peaks and then creates a window moving from peak to peak looking for the channel where this peak is real and adjust the intensities in the other channels.

Usage

```
pullup(mati, plotting=FALSE, channel=4)
```

Arguments

mati	matrix of intensities where each column is a channel/color for a given sample.
plotting	a TRUE/FALSE value indicating if the results from adjusting the intensities should be drawn or not.
channel	a numeric value indicating which of the channles/color (column) allocates the ladder intensties.

Details

No major details.

Value

If arguments are correctly specified the function returns:

mati A new matrix of DNA intensities corrected for overlapping of wavelenth readings in different channels.

References

Covarrubias-Pazaran G, Diaz-Garcia L, Schlautman B, Salazar W, Zalapa J. Fragman: An R package for fragment analysis. 2016. BMC Genetics 17(62):1-8.

Robert J. Henry. 2013. Molecular Markers in Plants. Wiley-Blackwell. ISBN 978-0-470-95951-0.

Ben Hui Liu. 1998. Statistical Genomics. CRC Press LLC. ISBN 0-8493-3166-8.

Examples

```
data(my.plants)
layout(matrix(1:2,2,1))
# without pull up adjustment
plot(my.plants[[1]][,1], type="l", col="blue", xlim=c(2750,2850))
lines(my.plants[[1]][,2], col="green")
lines(my.plants[[1]][,3], col="gold")
## adjusted
yy <- pullup(my.plants[[1]])
plot(yy[,1], type="l", col="blue", xlim=c(2750,2850))
lines(yy[,2], col="green")
lines(yy[,3], col="gold")
# general view
yy1 <- pullup(my.plants[[1]], plotting=TRUE)
```

---

read.abif *Read ABIF formatted files*

Description

ABIF stands for Applied Biosystem Inc. Format, a binary fromat modeled after TIFF format. Corresponding files usually have an .ab1 or .fsa extension.

Usage

```
read.abif(filename, max.bytes.in.file = file.info(filename)$size,
 pied.de.pilote = 1.2, verbose = FALSE)
```

Arguments

filename	The name of the file.
max.bytes.in.file	The size in bytes of the file, defaulting to what is returned by file.info
pied.de.pilote	Safety factor: the argument readBin is set as pied.de.pilote*max.bytes.in.file.
verbose	logical [FALSE]. If TRUE verbose mode is on.

Details

All data are imported into memory, there is no attempt to read items on the fly.

Value

A list with three components: Header which is a list that contains various low-level information, among which numelements is the number of elements in the directory and dataoffset the offset to find the location of the directory. Directory is a data.frame for the directory of the file with the number of row being the number of elements in the directory and the 7 columns describing various low-level information about the elements.

Author(s)

J.R. Lobry

References

Covarrubias-Pazaran G, Diaz-Garcia L, Schlautman B, Salazar W, Zalapa J. Fragman: An R package for fragment analysis. 2016. BMC Genetics 17(62):1-8.

Anonymous (2006) Applied Biosystem Genetic Analysis Data File Format. Available at http://www.appliedbiosystems.com/support/software_community/ABIF_File_Format.pdf. Last visited on 03-NOV-2008.

The figure in the example section is an attempt to reproduce figure 1A from:

Krawczyk, J., Goesmann, A., Nolte, R., Werber, M., Weisshaar, B. (2009) Trace2PS and FSA2PS: two software toolkits for converting trace and fsa files to PostScript format. *Source Code for Biology and Medicine*, **4**:4.

Examples

```
#No examples provided, please download seqinr package for going deeper in this function.
```

---

reals *Finding the real peaks*

Description

This function takes a list with the information of positions, heights and weights for an individual and using the panel information finds the real peaks by using the separate function and getting the tallest peaks in the confidence interval constructed for the heights in the inteval of interest.

Usage

```
reals(x, panel=c(100:400), shi=1, ploidy=2, left.cond=c(0.4,3),
      right.cond=0.2, window=0.5)
```

Arguments

x	List with 3 elements; the information of positions, heights and weights for an individual in certain channel
panel	A vector containing the base pair interval where the peaks should be searched for
shi	The number of base pairs to be used for discarding neighboring peaks to the tallest peaks, i.e. if 2 peaks are 0.3 bp together the smalles will be discarded
ploidy	A scalar value indicating the ploidy of the organism to be scored
left.cond	A percentage value indicating when peaks to the leaft of the tallest peaks should be considered real based on the height, i.e. a very close peak right before the tallest peak if smaller than the tallest (half the size of the tallest one will be real or not)
right.cond	A percentage value indicating when peaks to the right of the tallest peaks should be considered real based on the height, i.e. a very close peak right after the tallest peak if smaller than the tallest (half the size of the tallest one will be real or not)
window	A value in base pairs indicating how much is the error for detecting a peak in a sample that was provided in the panelas a real peak

Details

No major details.

Value

If arguments are correct the function returns a list containing

$pos the index positions for the intensities

$hei the intensities for the fragments found

$wei the putative weights in base pairs based on the ladder provided

References

Robert J. Henry. 2013. Molecular Markers in Plants. Wiley-Blackwell. ISBN 978-0-470-95951-0.

Ben Hui Liu. 1998. Statistical Genomics. CRC Press LLC. ISBN 0-8493-3166-8.

Examples

```
data(my.plants)
x <- big.peaks.col(my.plants[[1]][,1],100)#for any color
#reals(x, panel=c(260,280), shi=1, ploidy=2)
#still needs weight information in order to find the reals,
#works internally of score.easy function
```

`saturate` *Checking and correcting saturated peaks*

Description

This function takes a vector of intensities and looks for peaks above 8000 RFUs and correct for possible splits at the top of the peaks by inverting the vally between splitted peaks and correcting the peak.

Usage

```
saturate(y)
```

Arguments

y	a vector containing the DNA intensities for the capillary electrophoresis.

Details

No major details.

Value

If arguments are correctly specified the function returns:

$y A new vector of DNA intensities adjusted for saturated peaks over 8000 RFUs.

References

Robert J. Henry. 2013. Molecular Markers in Plants. Wiley-Blackwell. ISBN 978-0-470-95951-0.

Ben Hui Liu. 1998. Statistical Genomics. CRC Press LLC. ISBN 0-8493-3166-8.

Examples

```
data(my.plants)
y <- my.plants[[1]][,3]
layout(matrix(1:2,2,1))
plot(y, type="l", xlim=c(2750,2850))
y2 <- saturate(y=y)
plot(y2, type="l", xlim=c(2750,2850))
```

score.markers *Fragment analysis scoring*

Description

This function uses information from the fsa files read from storing.inds function and does the ssr calling in the channel specified and returns the index position, height and base pair position.

Usage

```
score.markers(my.inds, channel = 1, n.inds = NULL, panel=NULL, shift=0.8,
          ladder, channel.ladder=NULL,
          ploidy=2, left.cond=c(0.6,3), right.cond=0.35, warn=FALSE,
          window=0.5, init.thresh=200, ladd.init.thresh=200,
          method="iter2", env = parent.frame(), my.palette=NULL,
          plotting=TRUE,  electro=FALSE, pref=3)
```

Arguments

my.inds	List with the channels information from the individuals specified, usually coming from the storing.inds function output
channel	The channel you wish to analyze, usually 1 is blue, 2 is green, 3 is yellow, 4 is red and so on
n.inds	Vector specifying the plants to be scored
panel	A vector containing the base pair interval where the peaks should be searched for
shift	The number of base pairs to be used for discarding neighboring peaks to the tallest peaks, i.e. if 2 peaks are 0.3 bp together the smalles will be discarded
ladder	A vector containing the expected weights for the ladder peaks that will be found the using the find.ladder function
channel.ladder	A scalar value indicating in which channel or color the ladder was read
ploidy	A scalar value indicating the ploidy of the organism to be scored to decide the maximum number of peaks the program should look for. TO BE IMPLEMENTED SOON. STILL NOT FUNCTIONAL.
left.cond	A percentage value (0-1) indicating when peaks to the left of the tallest peaks should be considered real based on the height, i.e. a value of 0.5 would mean that a close peak (to the left of the tallest peak) will be picked only if such peak is at least 50 percent as tall with respect to the tallest peak. The second argument is the number of base pair indicating when peaks to the left of the tallest peaks should be considered real based on the distance, i.e. a value of 3 would mean that a close peak (to the left of the tallest peak) will be picked only if such peak is at least 3 base pairs far away from the tallest peak

`right.cond` A percentage value (0-1) indicating when peaks to the right of the tallest peaks should be considered real based on the height, i.e. a value of 0.5 would mean that a close peak (to the right of the tallest peak) will be picked only if such peak is at least 50 percent as tall with respect to the tallest peak.

`warn` A TRUE/FALSE value indicating if warnings should be provided when detecting the ladder

`window` A value in base pairs indicating how much is the error for detecting a peak in a sample when providing a panel with expected peaks.

`init.thresh` An initial value of intensity to detect peaks. We recommend not to deal to much with it unless you have highly controlled dna concentrations in your experiment.

`ladd.init.thresh` If samples were not sized using the info.ladder.attach function this value will be used to detect ladder peaks. Internally the program will use the `find.ladder` function. We recommend not to deal to much with it unless you identified special situations with your ladder

`method` If samples were not sized using the info.ladder.attach function this method will be used to detect ladder peaks. An argument indicating one of the 3 methods available; "cor" makes all possible combination of peaks and searches exhaustive correlations to find the right peaks corresponsding to the expected DNA weights, or "ci" constructing confidence intervals to look for peaks meeting the conditions specified in the previous arguments, "iter2" an iterative procedure looking for the most likely peaks meeting your ladder expectation. Default is "iter2".

`env` this is used to detect the environment of the user and load the result in the same environment. Don't mess with it please.

`my.palette` A character vector with the colors to be used when drawing the RFU plots. If NULL it will use the programmed palette.

`plotting` a TRUE/FALSE value indicating if the plots should be drawn or not. The default value is TRUE.

`electro` A TRUE/FALSE value indicating if the electrogram/gel should be drawn or not. The default value is FALSE.

`pref` A scalar value indicating how many plots should be drawn in the output plotting. The dafault is 3.

Details

Method "ci" has been depreciated, currently the method "iter2" is the default and uses the ladder provided and observed peaks to match them using an iterative procedure based on least squares.

Value

If arguments are correct the function returns a plot and a list containing

$pos the index positions for the intensities

$hei the intensities for the fragments found

$wei the putative weights in base pairs based on the ladder provided

References

We have spent valuable time developing this package, please cite it in your publication:

Covarrubias-Pazaran G, Diaz-Garcia L, Schlautman B, Salazar W, Zalapa J. Fragman: An R package for fragment analysis. 2016. BMC Genetics 17(62):1-8.

Robert J. Henry. 2013. Molecular Markers in Plants. Wiley-Blackwell. ISBN 978-0-470-95951-0.

Ben Hui Liu. 1998. Statistical Genomics. CRC Press LLC. ISBN 0-8493-3166-8.

Examples

```
## ================================= ##
## ================================= ##
## Fragment analysis requires
## 1) loading your data
## 2) matching your ladder
## 3) define a panel for scoring
## 4) score the samples
## ================================= ##
## ================================= ##

#####################
## 1) Load your data
#####################

### you would use something like:
# folder <- "~/myfolder"
# my.plants <- storing.inds(folder)
### here we just load our sample data and use the first 2 plants

?my.plants
data(my.plants)
my.plants <- my.plants[1:2]
class(my.plants) <- "fsa_stored"

######################
## 2) Match your ladder
######################

### create a vector indicating the sizes of your ladder and do the match

my.ladder <- c(50, 75, 100, 125, 129, 150, 175, 200, 225, 250, 275, 300, 325, 350, 375)
ladder.info.attach(stored=my.plants, ladder=my.ladder)

### matching your ladder is a critical step and should only happen once per batch of
### samples read

#############################################################################################
### OPTIONAL:
### If the ladder.info attach function detects some bad samples
### that you can correct them manually using
### the ladder.corrector() function
### For example to correct one sample in the previous data
```

```
### ladder.corrector(stored=my.plants,
#to.correct="FHN152-CPN01_01A_GH1x35_152-148-209_717-704-793_367-382-381.fsa",
#ladder=my.ladder)
##############################################################################################

#######################
## 3) Define a panel
#######################

### In fragment analysis you usually design a panel where you indicate
### which peaks are real. You may use the overview2 function which plots all the
### plants in the channel you want in the base pair range you want

overview2(my.inds=my.plants, channel = 2:3, ladder=my.ladder, init.thresh=5000)

### You can click on the peaks you think are real, given that the ones
### suggested by the program may not be correct. This can be done by using the
### 'locator' function and press 'Esc' when you're done, i.e.:
# my.panel <- locator(type="p", pch=20, col="red")$x
### That way you can click over the peaks and get the sizes
### in base pairs stored in a vector named my.panel

### Just for demonstration purposes I will use the suggested peaks by
### the program using overview2, which will return a vector with
### expected DNA sizes to be used in the next step for scoring
### we'll do it in the 160-190 bp region

my.panel <- overview2(my.inds=my.plants, channel = 3,
                    ladder=my.ladder, init.thresh=7000,
                    xlim=c(160,190)); my.panel

#########################
## 4) Score the samples
#########################

### When a panel is created is time to score the samples by providing the initial
### data we read, the ladder vector, the panel vector, and our specifications
### of channel to score (other arguments are available)

### Here we will score our samples for channel 3 with our panel created previously

res <- score.markers(my.inds=my.plants, channel = 3, panel=my.panel$channel_3,
                ladder=my.ladder, electro=FALSE)

### Check the plots and make sure they were scored correctly. In case some samples
### are wrong you might want to use the locator function again and figure out
### the size of your peaks. To extract your peaks in a data.frame do the following:

final.results <- get.scores(res)
final.results
```

`separate` *Separating peaks by a shift window*

Description

This function takes a list with positions, heights and weights called "g" and using a shift in base pairs determines when 2 neighboring peaks should be considered only one by getting the tallest peak. For example two peaks found at 173 and 173.5 base pairs are unlikely to be 2 different peaks, therefore only the tallest peak will pe chosen.

Usage

```
separate(g, shift=1, type="bp")
```

Arguments

`g`	List with 3 elements; the information of positions, heights and weights for an individual in certain channel
`shift`	The number of base pairs to be used for discarding neighboring peaks to the tallest peaks, i.e. if 2 peaks are 0.3 bp together the smalles will be discarded
`type`	A word indicating if the shift to be used should be used in base pairs or in index. The use is "bp" or "ind"

Details

No major details.

Value

If arguments are correct the function returns a new list containing

$pos the index positions for the intensities

$hei the intensities for the fragments found

$wei the putative weights in base pairs based on the ladder provided

References

Covarrubias-Pazaran G, Diaz-Garcia L, Schlautman B, Salazar W, Zalapa J. Fragman: An R package for fragment analysis. 2016. BMC Genetics 17(62):1-8.

Robert J. Henry. 2013. Molecular Markers in Plants. Wiley-Blackwell. ISBN 978-0-470-95951-0.

Ben Hui Liu. 1998. Statistical Genomics. CRC Press LLC. ISBN 0-8493-3166-8.

Examples

```
data(my.plants)
x <- big.peaks.col(my.plants[[1]][,1],100)#for any color
#separate(x, shift=1, type="bp") #still needs weight information
```

storing.inds *Extracting channel information*

Description

This function reads the FSA files using a function named 'read.abif' from another R package called seqinr. This will extract the information of the DNA intensities of the capillary electrophoresis and will store it in a data structure know in R as a list. The usage of the function and the arguments it takes are as follows:

Usage

```
storing.inds(folder, channels=NULL, fourier=TRUE,
            saturated=TRUE, lets.pullup=TRUE,
            plotting=FALSE, rawPlot=FALSE,
            llength=3000, ulength=80000)
```

Arguments

folder	A path/directory where the FSA files are located. We recommend to use the Seession tab -> Set working directory and provide that folder as an argument
channels	A scalar value indicating how many channels/colors should be found in the FSA files, usually people using the rox375 ladder in the red channel have 4 channels, people using the LIZ ladder have 5 channels. The default is the last channel.
fourier	A FALSE/TRUE value indicating if data should be smooth aplying a Fourier transormation using 40 percent of the lowest frequencies. The dafault is TRUE
saturated	A FALSE/TRUE value indicating if data should be checked and treated for saturated peaks above 8000 RFU which usually split at the top in 2 different peaks. The dafault is TRUE
lets.pullup	A FALSE/TRUE value indicating if data should be treated for noise from channel to channel known as pull up or pull down peaks since wavelengths where the dyes are read usually overlap (blue->green->yellow->red->orange. The dafault is TRUE
plotting	A FALSE/TRUE value indicating if results after data cleaning steps should be plotted to asses graphically how data was handled. The dafault is FALSE
rawPlot	A FALSE/TRUE value indicating if a plot drawing all vectors read should be plotted. The dafault is FALSE since this consumes a lot of memory.
llength	A numeric value indicating how small can be the number of indexes in each channel. Default is 3000 indexes for small runs.
ulength	A numeric value indicating how big can be the number of indexes in each channel. Default is 80000 indexes for long runs.

Details

No major details.

Value

If arguments are correct the function returns a list containing

all.inds.mats A list where each element is a data frame containing the "n" channels of an individual

References

We have spent valuable time developing this package, please cite it in your publication:

Covarrubias-Pazaran G, Diaz-Garcia L, Schlautman B, Salazar W, Zalapa J. Fragman: An R package for fragment analysis. 2016. BMC Genetics 17(62):1-8.

Robert J. Henry. 2013. Molecular Markers in Plants. Wiley-Blackwell. ISBN 978-0-470-95951-0.

Ben Hui Liu. 1998. Statistical Genomics. CRC Press LLC. ISBN 0-8493-3166-8.

Examples

```
data(my.plants)
### the correct way to do it for a population of inds with
### 4 colors + ladder= 5 would be:
# my.plants <- storing.inds(folder)
```

---

threshs *Customizing thresholds*

Description

This function takes data contained in a list with 2 elements, the first containing the position in base pairs and the heights of the positions and given a panel finds the best minimum threshold by creating a confidence interval in order to get only the real peaks. Implemented internally of auto,score and score.easy function.

Usage

```
threshs(my.plant, min.thre=200, panel, ci=1.9)
```

Arguments

`my.plant`	List with 2 elements; the position in base pairs and the heights associated with them
`min.thre`	Starting threshold to find the peaks and refine the search
`panel`	Two scalar values indicatin the interval where the peaks should be found
`ci`	Scalar value indicating the number of standard errors to be used when creating the confidence interval for ssr calling

Details

No major details.

Value

If arguments are correctly specified the function returns a list containing

$newthre A scalar value indicating the new threshold to be used when calling the real peaks

References

Robert J. Henry. 2013. Molecular Markers in Plants. Wiley-Blackwell. ISBN 978-0-470-95951-0.

Ben Hui Liu. 1998. Statistical Genomics. CRC Press LLC. ISBN 0-8493-3166-8.

Examples

```
data(my.plants)
# implemented internally of auto,score and score.easy function
# threshs(my.plant, min.thre=200, panel=(260,290), ci=1.9)
```

---

transfft *Applying the fourier transformation to a data frame*

Description

This function takes a vector and applies a fourier transformation in order to smooth the peaks usinf the fft function in the base package. Use only top 40 percent of the lowest frequencies.

Usage

```
transfft(sn, top=0.3)
```

Arguments

sn	a numeric vector containing the DNA intensities for the capillary electrophoresis.
top	percent of lowest frequencies that should be used for the fourier transformation.

Details

No major details.

Value

If arguments are correctly specified the function returns:

$y A new vector of DNA intensities smoothed to avoid extra noisy peaks.

References

Covarrubias-Pazaran G, Diaz-Garcia L, Schlautman B, Salazar W, Zalapa J. Fragman: An R package for fragment analysis. 2016. BMC Genetics 17(62):1-8.

Robert J. Henry. 2013. Molecular Markers in Plants. Wiley-Blackwell. ISBN 978-0-470-95951-0.

Ben Hui Liu. 1998. Statistical Genomics. CRC Press LLC. ISBN 0-8493-3166-8.

Examples

```
data(my.plants)
g1 <- transfft(my.plants[[1]][,4], top=0.8)
g2 <- transfft(my.plants[[1]][,4], top=0.4)
g3 <- transfft(my.plants[[1]][,4], top=0.1)
layout(matrix(1:3,3,1))
plot(g1, type="l")
lines(g2, col="red")
lines(g3, col="blue")
par1 <- c("top=0.8", "top=0.4", "top=0.1")
par2 <- c("black", "red", "blue")
par3 <- c(1,1,1)
legend("topright", legend=par1, col=par2, bty = "n", lty=par3, lwd=par3, cex=0.75)
```

---

transp *Creating color with transparency*

Description

This function takes a color and returns the same with a certain alpha grade transparency.

Usage

```
transp(col, alpha=0.5)
```

Arguments

col	Color to be used for transparency
alpha	Grade of transparency desired

Details

No major details.

Value

If arguments are correctly specified the function returns:

$res A new color with certain grade of transparency

References

Covarrubias-Pazaran G, Diaz-Garcia L, Schlautman B, Salazar W, Zalapa J. Fragman: An R package for fragment analysis. 2016. BMC Genetics 17(62):1-8.

Robert J. Henry. 2013. Molecular Markers in Plants. Wiley-Blackwell. ISBN 978-0-470-95951-0.

Ben Hui Liu. 1998. Statistical Genomics. CRC Press LLC. ISBN 0-8493-3166-8.

Examples

```
transp("red", alpha=0.5)
```

Index